AFRL-IF-RS-TR-2004-299
Final Technical Report
October 2004

DISTRIBUTED INFORMATION ENTERPRISE MODELING AND SIMULATION (DIEMS)

CACI Technologies, Inc.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** October 2004	**3. REPORT TYPE AND DATES COVERED** **FINAL** **Sep 01 – Aug 04**

4. TITLE AND SUBTITLE

DISTRIBUTED INFORMATION ENTERPRISE MODELING AND SIMULATION (DIEMS)

5. FUNDING NUMBERS

C - F30602-00-D-0221, Task 10
PE - 62702F
PR - SOS1
TA - QF
WU - 12

6. AUTHOR(S)

Gary Blank

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

CACI Technologies, Inc.
1300 Floyd Avenue
Rome NY 13440

Metron, Inc.
512 Via De La Valle, Suite 301
Solana Beach CA 92075

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFTC
26 Electronic Parkway
Rome NY 13441-4514

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2004-299

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: James Hanna/IFTC/(315) 330-3473 James.Hanna@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

The Air Force Research Laboratory (AFRL) has produced a communication simulation called DIEMS (Distributed Information Enterprise Modeling and Simulation) that is based on Metron's parallel simulation framework, SPEEDES (Synchronous Parallel Environment for Emulation and Discrete-Event Simulation). Over the course of the contract, Metron has assisted AFRL in three broad categories of activity: (1) providing the capabilities to allow DIEMS to participate in High-Level Architecture (HLA) federations; (2) general support implementing, testing and debugging DIEMS; and (3) upgrades to the SPEEDES framework. Metron has assisted in developing and debugging DIEMS, in all aspects of integrating DIEMS into an HLA federation, and provided improvements to the SPEEDES framework. Because of the challenging nature of parallel simulation, support of this kind is an extremely valuable resource. By increasing the rate of development and preventing snags from turning into lengthy delays, it has been a wise investment.

14. SUBJECT TERMS

Information enterprise, enterprise modeling, distributed simulation, high performance computing, SPEEDES, HLA, FOM, SOM

15. NUMBER OF PAGES 22

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

1.0 INTRODUCTION

AFRL has produced a communications simulation called DIEMS (Distributed Information Enterprise Modeling and Simulation) that is based on Metron's parallel simulation framework, SPEEDES (Synchronous Parallel Environment for Emulation and Discrete-Event Simulation). Over the course of this contract, Metron has assisted AFRL in three broad categories of activity:

1) Providing the capabilities to allow DIEMS to participate in High-Level Architecture (HLA) federations,

2) General support implementing, testing, and debugging DIEMS, and

3) Upgrades to the SPEEDES framework.

We will summarize the help provided by Metron to AFRL.

2.0 HLA FEDERATIONS

A High Level Architecture (HLA) federation is a group of simulations lashed together in order to create a larger and/or more detailed simulation. For example, an air warfare simulation could be combined with naval and land warfare simulations to form a more comprehensive warfare simulation. Or, the air warfare simulation could be augmented with a communications simulation to provide more detailed modeling of air warfare scenarios. The software framework that allows the individual simulations (called federates) to communicate and coordinate is called the HLA Runtime Infrastructure (HLA RTI or simply RTI).

With respect to the task of allowing DIEMS to participate in HLA federations, Metron's responsibilities were:

1) Federation Object Model (FOM) Development Support

The FOM specifies the precise details of the interface through which federates will communicate. This is the essential element that allows totally different simulations to work together. For example, by specifying exactly what attributes (and their types) make up an Aircraft object, simulation A can be notified about such objects in simulation B, and can comprehend the evolving state of B's Aircraft objects during the federation execution. This is because federates A and B have agreed upon what constitutes an Aircraft object and have specified this in the FOM. In addition to objects, the other major class of entity specified in the FOM is interactions. By precisely specifying the name and parameters associated with a particular interaction, the FOM allows simulation A to affect simulation B by sending it an interaction. Metron's task was to provide assistance to AFRL in formulating the FOM to be used in the federation.

2) DIEMS HLA Compliance support

In order to participate in an HLA federation, a federate must be able to interface correctly with the RTI. The basic elements involved in this "handshaking" include: create/join/resign from federation executions; publish/subscribe to object classes and interactions; send/receive attribute updates and interactions; and coordinate with HLA time management. Metron's goal was a methodology that is convenient, efficient, general, reliable, and which has minimal impact on DIEMS's code.

In addition to providing this interface between DIEMS and the RTI, Metron would also provide a test federation with which to verify that DIEMS could perform the necessary functions required of a federate.

3) Federation Integration Support

Metron's experience in other HLA federations indicates that one should expect to encounter a certain number of difficulties when the federates finally get together and attempt to communicate/coordinate in an HLA federation. Thus, Metron’s task was to work with the federation members to help identify and fix problems related to the inclusion of DIEMS in the federation.

2.1 GIEsim

The idea of federating DIEMS finally became a reality with the creation of the Global Information Enterprise Simulation (GIEsim) federation in September of 2003. The process of assembling this federation began about a year prior to that point, with Metron providing assistance of the sort described above. In the following, we describe the specific activities Metron performed over time to help the DIEMS group integrate their simulation into the GIEsim federation.

<u>September 2002</u>

1) On 18 September, a representative from Metron attended a meeting at Rome Labs to discuss plans for GIEsim. He gave a presentation outlining Metron's role in these activities. In addition to providing support for DIEMS (which will be used in GIEsim), Metron's main tasks for the immediate future are as follows:
    a) Design the interfaces required to combine DIEMS with OPNET in an HLA federation. OPNET is a simulation that does highly detailed modeling of communications networks. The intent was for DIEMS to use OPNET to simulate transmission times of messages sent from one location to another.
    b) Investigate the feasibility of using communications output from Metron's Naval Simulation System (NSS) as input to drive a DIEMS scenario.

2) Metron began work defining the interfaces mentioned in item 1a) above. We have requested guidance from AFRL in order to determine which capabilities in OPNET would best complement DIEMS. This information is needed to specify the HLA interface for the DIEMS/OPNET federation (i.e. the Simulation Object Models [SOMs] and the Federation Object Model [FOM]). The FOM contains a list of public objects and interactions through which the simulations can communicate.

In the meantime, Metron has begun investigating the specifics of how DIEMS will utilize the SPEEDES HLA Gateway module in order to enable a multi-CPU execution to act as a single HLA federate. The HLA Gateway contains the code to allow DIEMS to join an HLA federation execution, publish and subscribe to object and interaction classes, send and receive attribute updates and interactions, etc.

October 2002

The majority of Metron's effort was spent investigating how to interface DIEMS with OPNET. One possibility being considered is to use the High Level Architecture (HLA) Runtime Infrastructure (RTI), specifically the "Next Generation" version (RTI-NG) produced by SAIC. This RTI was installed and tested (fortunately, just prior to the end of DMSO's free-download program). In order to work with this software, it was necessary to obtain a newer version of the GNU g++ compiler (version 3.0.2). This compiler was installed and used to build the SPEEDES HLA Gateway, which is the conduit through which DIEMS could work with the RTI-NG. This required dealing with a fair number of compatibility issues and other problems.

In the meantime, a teleconference was arranged between Metron and AFRL in order to clarify the immediate goals of this task (and also the task of investigating the feasibility of using communications output from Metron's Naval Simulation System (NSS) as input to drive a DIEMS scenario). At this meeting, it was revealed that a group at Wright-Patterson was producing a smaller, simpler, HLA-like interface for OPNET. We will need to decide whether to use an altered version of the HLA Gateway to work with this OPNET interface, or else build a new SPEEDES interface customized for this purpose.

November 2002

In the beginning of the month, a teleconference was held in order discuss Metron's tasking to support DIEMS and GIEsim. During this meeting, AFRL personnel clarified the immediate tasks that Metron would fulfill. These consisted of the following two items:

1) Create a design for federating DIEMS with OPNET and prepare a presentation (for the 12/4 GIEsim meeting) describing the design. The "Comm API" interface will be used to connect the two simulations. This is a small, socket-based, inter-simulation framework developed by AFRL at Wright-Patterson AFB. It contains basic message passing utilities and an HLA-like time management function.

2) Investigate the feasibility of using Metron's Naval Simulation System (NSS) to generate input scenarios to test DIEMS, and prepare to present the findings at the 12/4 GIEsim meeting. Our task was to decide whether this is a practical approach, and if so, design a procedure for producing these scenarios.

Throughout November, Metron researched the above two topics and prepared a PowerPoint presentation describing the results. For the first task, we obtained a paper from Lt. Dooley describing the OPNET Comm API interface that he (and others) had built. Then we considered what would be the best way to use the Comm API to manage communications between DIEMS and OPNET. We decided the simplest and most efficient design would be a SPEEDES external module, with the Comm API code linked in. The other main alternative was to "retrofit" the existing HLA Gateway so that it could work with the Comm API in lieu of an HLA RTI. This idea was discarded because the HLA Gateway is unnecessarily large and complex given the simplicity of the Comm API. In addition to this, elaboration of the time management mechanism was accomplished.

In order to do the second task, several meetings were arranged with NSS programmers and analysts in order to determine what could reasonably be extracted from NSS output and/or input. The idea was to see if there was enough information available with which to construct DIEMS input scenarios. In addition to this, Metron downloaded a current version of DIEMS and analyzed the input files from which it builds its scenario. When these inputs were fully understood, a list was made comparing what DIEMS requires as input with information that could somehow be gotten from NSS output and/or input. This comparison was the basis for determining the practicality of using NSS to generate DIEMS input scenarios.

<u>December 2002</u>

On 4 December, the GIESim Final review meeting was held at AFRL facilities in Rome, NY. A Metron representative attended and gave a presentation on the following two topics:

1) Metron's design for federating DIEMS with OPNET using AFRL's Comm API interface to coordinate the two simulations.
2) Metron's assessment of the feasibility of using the Naval Simulation System (NSS) to help generate scenarios to test DIEMS.

The results, in a nutshell, were as follows:

1) The federation design will be based on a SPEEDES external module linked to the Comm API software.
2) Although NSS could provide some valuable inputs needed to drive DIEMS, there is enough of a mismatch between NSS and DIEMS to render this approach too difficult to be generally practical.

<u>May 2003</u>

After some debate, AFRL decided that DIEMS would participate as an HLA federate in an upcoming GIEsim federation. Metron has been investigating how best to interface DIEMS with the GIEsim federation. Possibilities under consideration are:

1) The SPEEDES HLA Gateway,
2) A simplified external module HLA interface, and
3) A simulation object that directly interfaces with the HLA RTI.

<u>June 2003</u>

The initial stages of assembling the first GIEsim federation have started and Metron began the process of integrating DIEMS into this federation. As was stated in the May 2003 monthly report, the three options for interfacing DIEMS with the HLA RTI are as follows:

1) The SPEEDES HLA Gateway,
2) A simplified external module HLA interface, and
3) A simulation object that directly interfaces with the HLA RTI.

To ensure that an interface would be available on time, Metron took a two-pronged approach, with one programmer investigating option 1) and another investigating options 2) and 3). Due to the size and complexity of the HLA Gateway (and other serious problems), option 1) was eliminated. Option 3) was eliminated because of the difficulty of directly coordinating time management between SPEEDES and the HLA RTI. Thus, option 2) was selected.

An external module HLA interface has been built. Since the biggest hurdle is doing the time management, that task was attacked first. Metron soon accomplished this and then proceeded to build interfaces for sending and receiving HLA interactions (since this is the main requirement for this federation). Work continues on this HLA interface. Our goals are to: 1) make the interface as simple as possible to use (while minimizing the impact on DIEMS code), 2) build in as much general capability as possible, and 3) keep the interface code simple and efficient.

In addition to the above, Metron has been communicating with other federation members in order to be informed about plans for the GIEsim federation.

<u>July 2003</u>

During July, the finishing touches were put on the software that will be used to interface between DIEMS and the HLA runtime infrastructure (RTI). This is a separate program based on the SPEEDES external module concept. Because of this, it can:

1) Control the time advance of DIEMS,
2) Receive messages from DIEMS, and
3) Insert events into DIEMS.

In addition to the above, it has been designed and tested with the version of the RTI that will be used in this federation. Since it handles much of the RTI-specific requirements (e.g. create/join/resign from federation execution, time management, specifics of publishing, subscribing, sending and receiving interactions, etc.), the impact on DIEMS code will be minimized. Although built with this federation in mind, the interface software has been designed to be general-purpose in order to streamline integration in subsequent GIEsim federations.

In addition to the above, Metron has consulted with other federation members so as to insure that we understand the nature of the role DIEMS is to play in the federation.

<u>August 2003</u>

During August, Metron helped prepare for the approaching integration of DIEMS into the first GIEsim federation. This involved three types of activities:

1) Communicating with other federation members in order to understand the precise nature of how DIEMS will have to interface with the federation, and
2) Making sure our interface can handle these requirements,
3) Creating an example federation to test the SPEEDES/RTI interface and to illustrate how the software can be used by DIEMS to participate in the GIEsim federation.

With regard to item 1), SAIC at Wright-Patterson AFB was very helpful in providing the interaction specifications (the .fed and .omd files) and in describing how time management and initial synchronization would be handled. The lack of logical time management in the federation (federates will synchronize by running at wall clock speed) presented a problem for our interface, which assumes that the federate will be time regulating and time constrained. Although SPEEDES can be made to run at wall clock speed, it was decided that an easier solution would be to have at least one other time regulating/constrained federate in order to make our logical time management meaningful. Lacking this, all time advance requests would be immediately granted, which means that DIEMS would simply run as fast as possible.

After learning about the requirements for the federation, we created an example federation intended to mimic what DIEMS will need to do in the GIEsim federation. This consisted of a SPEEDES-based federate, a non-SPEEDES federate, and the SPEEDES/RTI interface. The software was packaged up and emailed to AFRL, along with instructions on how to make/run the executables, and an explanation of how to send/receive HLA interactions and how to pack/unpack the interaction parameters. An updated version of this federation was sent to AFRL two days later. This version contained the actual interactions (and parameters) that DIEMS will need to send/receive in the GIEsim federation, the intent being that AFRL programmers will be able to use code similar (or identical) to the example code in DIEMS.

3.0 DIEMS SUPPORT

In addition to the other activities described, much of the assistance Metron provided to AFRL falls under the heading of general support of the development of DIEMS. Despite the high-level capabilities of AFRL's programmers, the simple fact is that parallel simulation is an extremely challenging business, especially for beginners. The task of correctly managing a simulation distributed over an arbitrary number of processors, with events executing concurrently (and optimistically), is a very tricky one, with many pitfalls.

Perhaps the most obvious example is the need to make events rollbackable. Since SPEEDES is an optimistic framework, it is possible for an event to be processed on an object which subsequently receives an event with an earlier timestamp, *t*. To correctly execute the latter event, the state of the object must be *rolled back* to time *t* before processing the event. Although SPEEDES provides data types and functions for doing this, it is somewhat conceptually difficult, and it is easy to create subtle bugs by forgetting to make *all* of the object state rollbackable.

Since Metron's staff has, in aggregate, many years of experience working with SPEEDES-based simulations, they can often root out these (and other) bugs quickly, thus saving AFRL developers valuable time. Metron also has furnished various other support services, such as providing example software demonstrating how to use SPEEDES features (e.g. the external module). Because this support accelerates development and improves simulation correctness, we believe this to be an investment that pays large dividends. In the following sections, we describe some of the support functions provided by Metron.

<u>January 2002</u>

After overcoming some obstacles, Metron was able to compile and run AFRL's DIEMS simulation. One of the major problems was the fact that a software package used by DIEMS (called "Xerces") required C/C++ libraries that were incompatible with the version of Linux used at Metron. This problem was solved by building the Xerces library from the source.

When we got the simulation running, we noticed several warning messages in the output. By tracking down the source of these, a bug was discovered in the code, a string that was not NULL-terminated. When we checked out a later version of the DIEMS, we noticed that someone at AFRL had already fixed this problem.

Next, DIEMS was linked with SPEEDES version 2.0.1 in order to find out why the two were incompatible. We soon discovered the cause, a change in the way SPEEDES handles iterators (specifically `RB_SpBinaryTree` iterators). After analyzing the situation, we decided the best way to fix this problem was to add two methods to the `RB_SpBinaryTree` and `SpBinaryTree` classes. These would clearly and efficiently handle the case in which one wishes to remove the current item during an iteration over a binary tree object.

Metron implemented the fix and verified that DIEMS could now run with the newer version of SPEEDES. This situation was explained to AFRL in an email. AFRL should be able to upgrade to SPEEDES 2.0.1 without much difficulty when convenient to do so.

<u>February 2002</u>

Metron provided assistance locating and fixing difficulties in DIEMS. This included the following:

1) AFRL reported an intermittent crash to Metron. The cause was traced to a bug in SPEEDES verion 1.1e (which was fixed in version 2.0.1). A patch file was sent to AFRL to deal with this problem until they upgrade to version 2.0.1.

2) While trying to determine the source of incompatibility between DIEMS and SPEEDES version 2.0.1, Metron reported two potential bugs. Fixes were coded up by Metron personnel and checked into AFRL's source code repository.

3) While testing the code in 2), we noticed that DIEMS occasionally crashed. This led us to find five possible problems in the code. AFRL was notified about these issues and was given recommendations specifying how to fix them.

4) Metron contacted AFRL to discuss problems arising from the use of SPEEDES iterators. Metron proposed a solution intended to avoid possible confusion and bugs, and which will make iterators more efficient.

<u>April 2002</u>

Metron completed the coding and testing of a new SPEEDES release (2.0.1e) for use by AFRL. The major upgrades were as follows:

1) Iteration

SPEEDES 2.0.1e has new methods for removing items from container classes while iterating through the container. These are:

`void *RemoveCurrentElementAndGoToPrevious()` and
`void *RemoveCurrentElementAndGoToNext()`.

These remove the current element, move the internal or external iterator to the previous/next element in the iteration sequence, and return a `void*` pointer to the element that was just removed (e.g. so it can be deleted).

These two methods were implemented in the following classes (for internal iteration):

`SpList`, `RB_SpList`, `SpBinaryTree`, `RB_SpBinaryTree`, `SpHashTree`, and `RB_SpHashTree`.

They were also implemented for the following (external) iterator classes:

`SpIterator_SpList`, `SpIterator_RB_SpList`, `SpIterator_SpBinaryTree`, `SpIterator_RB_SpBinaryTree`, `SpIterator_SpHashTree`, and `SpIterator_RB_SpHashTree`.

2) MPI Capabilities

SPEEDES libraries can now be built such that communications are performed using MPI calls (rather than the SPEEDES shared memory communications library). This can be done to improve portability.

3) Data Distribution Management (DDM) fixes/upgrades.

4) Miscellaneous bug fixes

AFRL wishes to upgrade to SPEEDES 2.0.1e as soon as possible. Toward this end, Metron helped locate and fix the incompatibilities between DIEMS and SPEEDES 2.0.1e. This entailed the following tasks:

1) Checking out a new version of DIEMS.
2) Getting DIEMS to run with SPEEDES 2.0.1e. This required changing DIEMS code to use the new `RemoveCurrentElementAndGoToNext` methods.
3) Running DIEMS's regression test suite to verify that the simulation still works as expected.

Metron investigated a discrepancy involving test D_008, and found a bug in the test. This was caused by an event that performed non-rollbackable operations on a list contained in a simulation object. Metron fixed the problem and sent an explanation to AFRL.

<u>June 2002</u>

Metron completed a basic simulation/external module system intended to provide an example demonstrating how to efficiently transfer large blocks of data from a simulation to an external module. This capability is required by DIEMS. The code and a detailed description were emailed to AFRL. This example should help streamline this portion of DIEMS development by providing a model on which to base AFRL's code.

<u>July 2002</u>

1) In June, Metron delivered a basic simulation/external module system intended to provide an example demonstrating how to efficiently transfer large blocks of data from a simulation to an external module. AFRL developers used this to implement a similar capability in DIEMS, but ran into a problem: the external module and DIEMS did not synchronize at the end of the simulation as expected.

   AFRL reported the problem and sent the code to Metron by e-mail. The problem was reproduced at Metron. After analyzing this difficulty, it was determined that the best solution was a small patch of the SPEEDES code. This patch insures that all messages sent from the simulation are handled by the external module, even if the simulation has already completed. Thus, there is no longer any need for the external module and DIEMS to synchronize at the end of the simulation, which simplifies the code. Metron sent this patch and a number of files containing modifications to the DIEMS/external module code.

2) AFRL requested a "clean" copy of SPEEDES version 2.0.1e. Metron sent this and a slightly modified version of 2.0.1e. The latter contained the patch mentioned above, as well as some minor alterations needed because of changes in newer version of the g++ compiler.

August 2002

AFRL reported a difficulty working with an external module attached to a DIEMS simulation. The problem was the inability to invoke a simulation event from the external module. Metron looked into this issue using a copy of the code located at its Solana Beach facilities. In general, there was no difficulty invoking events from the external module, but the following problem was identified: it is possible for the simulation to quit before all external messages have arrived (and been handled). This probably explains the difficulties experienced at AFRL.

Metron began analyzing the cause of this problem and started investigating possible solutions. It will probably be necessary to reinstate the end-time synchronization in order to ensure that DIEMS does not end before the external module has completed all communications with the simulation.

Metron e-mailed an explanation to AFRL, describing how to invoke simulation events from an external module. This e-mail also asked how AFRL intended to use this capability so that we could try to devise the solution best suited to AFRL's requirements.

December 2002

The major issue addressed had to with difficulties experienced by AFRL coordinating DIEMS with a SPEEDES external module attached to DIEMS. One specific problem was the apparent inability to schedule events from an external module into DIEMS. AFRL provided an example in which commands sent to DIEMS from an external module did not appear as events in the simulation. The code was compiled and executed at Metron and the behavior was duplicated.

At this point, our task was to determine why the given external module was unable to schedule events in DIEMS. After some debugging, the cause of this particular problem was found. However, because the correspondence from AFRL indicated a need to be able to access the full capabilities of external modules, the task was expanded beyond explaining the source of this specific difficulty. A series of cases involving the interaction between a simulation and an external module were coded up and tested. These investigated the details of such things as: the exact implications of the speedes.par *lookahead* parameter; the effects of the `SpStateMgr`'s time lag; how the timestamp of an event generated by a `SendCommand` call in an external module is determined; accessing the `SpEmHostUser`'s `ScheduleEvent` method in order to specify the simulation time of an event generated by an external module; using named pauses to be able to schedule events arbitrarily close to the simulation end time; the determination of the timestamp of external module events generated by a simulation; and precisely defining the implications of invoking the `SpStateMgr`'s `GoToTime` call. Notes were gathered on the specifics of all the above. The intention is to send AFRL programmers' e-mails clarifying the use of external modules so that they can make use of these capabilities with greater ease.

January 2003

Metron investigated various aspects of the use of external modules, and then summarized the information in e-mails to AFRL programmers. The intention is to clarify certain complex issues and to help AFRL more quickly and effectively utilize the SPEEDES external module capabilities. Two e-mails were sent and a third is nearly completed.

The first e-mail dealt with the following items:

1) An explanation why AFRL's example code was unable to schedule events into DIEMS from the external module,

2) Defining when the external module receives messages from DIEMS,
3) The meaning and effects of setting the speedes.par *lookahead* parameter to a positive value,
4) A description of the relationship between the simulation time and the external module time, and
5) A detailed explanation of issues associated with scheduling events from the external module into a SPEEDES simulation.

The second e-mail addressed complicated timing issues and included three PowerPoint slides to help illustrate the explanations. Specifically, it dealt with:

1) The time interval in which events generated (from an external module) by `SendCommand` will be scheduled in the simulation,
2) Using `ScheduleEvent` (where possible) to specify the exact time of an event scheduled by the external module, and
3) Determining the minimum worst-case response time for an external module reacting to a SPEEDES event.

The third e-mail is mostly written and will be sent shortly. It addresses a specific question sent to Metron by AFRL: how to use the external module to specify which data is to be sent out from DIEMS. This includes a fair bit of example code defining a message class and sketching out how to send and receive these messages.

<u>February 2003</u>

Metron completed the third and final e-mail clarifying various issues related to the use of SPEEDES external modules. This e-mail described how to tell each simulation object of a given type what information to send to the external module. Highlights included the following:

1) A code example showing how to schedule an event (from an external module) on all instances of a given type of simulation object.
2) An example message class for sending information requests to `S_Platform` objects. The example shows how to use SPEEDES `NET_INT` and `NET_DOUBLE` types to insure that integers and doubles are properly communicated between big- and little-endian architectures.
3) A description of the use of a SPEEDES named pause to send the above messages to `S_Platform` objects before the start of the simulation.

In addition to the above, Metron investigated a number of porting issues, including the following:

1) Transitioning to g++ version 3.0 (and later) compilers. Metron programmers have already made SPEEDES compatible with g++ version 3.2. A compatible release of SPEEDES is planned for July.
2) A recommendation of Major Shared Resource Center (MSRC) computers on which to port DIEMS. In addition to running DIEMS on a Linux cluster and a Sun Enterprise (which AFRL has evidently already done), our next two choices are SGI and Compaq HPCs.
3) An investigation of SGI-related porting problems. Metron compiled and linked DIEMS on an SGI computer to try to identify compatibility problems between DIEMS code and SGI compilers and/or libraries (especially STL).

<u>March 2003</u>

In March, Metron began the task of porting DIEMS to Major Shared Resource Center (MSRC) high-performance computers (HPCs). To do this, the first step was to obtain an account allowing access to various MSRC machines. This involved some administrative tasks (e.g. filling out and sending required forms, determining the permissibility of loading SPEEDES and DIEMS onto MSRC machines, etc.) and a fair bit of technical work, such as the following:

1) Downloading, installing and learning about the Kerberos software package. This is a security system designed to authenticate users attempting to access MSRC computers, and also to encrypt communications. An older version of Kerberos used at Metron was unable to work from behind a firewall and therefore had to be installed and run from a machine outside the firewall. Because of this, we were uncertain whether the new version of Kerberos could function behind Metron's firewall. After initial attempts to use Kerberos from behind the firewall failed, it was installed on a computer located outside the firewall. This machine had an incompatible version of the Linux operating system, so the software would not work. This turned out to be a blessing in disguise because it forced us back to the problem of getting Kerberos to work from behind the firewall. With some prompt assistance from the MSRC Helpline people, this problem was soon solved. The fix was then recorded and automated using Unix aliases.

   After getting Kerberos to work, we logged onto a few of the MSRC computers and dealt with a few technical difficulties. Then SPEEDES was uploaded and compiled on one of the SGI HPCs.

2) DIEMS was quickly uploaded, but building it has presented a number of difficulties. To begin with, the SGI machines lacked the software tools required by the DIEMS installation procedure: autoconf, automake, aclocal, and autoheader were not available. This was odd because other GNU software was installed on these machines, but we searched and could not locate these tools (possibly the reason for this is that autoconf, etc. are used for installing software, which is generally discouraged by MSRC). The solution to this problem (the lack of autoconf, etc.) was as follows:

   a) By running autoconf on a Linux machine at Metron, we produced the "configure" script and three "Makefile.in" files.

   b) The configure script and the Makefile.in files were uploaded to the SGI machine.

   c) The configure script was then executed on the SGI machine.

Another possible solution is to install the missing tools. We want to find the best way to do this since AFRL intends for DIEMS to be easily portable to other machines besides Linux computers.

Finally we were able to investigate the cause of the compilation problems experienced by AFRL when using SGI machines. We began working with the GNU compiler (g++), but switched to SGI's compiler because there were header files missing from the GNU installation. Because AFRL has told us that they prefer g++ (since it will probably require fewer changes in DIEMS code), we may have to install g++ ourselves on one of the SGI HPCs. In the meantime, we have been using SGI's compiler, which has the benefit of producing lots of warning messages. One of these alerted us to a problem in a section of new DIEMS code.

Disregarding the compiler for the time being, Metron has discovered the major (and possibly only) cause of the compilation problems: the changing C++ standard in which globally-defined entities (such as `ostream`, `cout`, `cerr` and many others) are being transitioned to the `std` namespace. Currently, there are two definitions of these entities: a global one and one specified inside the `std` namespace. The essence of the problem is this: because both sets of definitions are being used, and because the `std` names are being exported into the global namespace (via `using namespace std;`), there are multiple definitions of the same name (e.g. `cout`, `ostream`, `cerr`, etc.) in the global namespace. Since the compiler cannot resolve which is the desired definition, it flags all such references as errors.

This is a tricky problem. Metron is currently investigating different ways of solving it. The tendency to import all `std` names into the global namespace (with `using namespace std;`) probably should be avoided. The `std` namespace is a repository that will grow over time. Polluting the global namespace with the entire contents of `std` is a ticking time bomb that can break the code by introducing name conflicts. On the other hand, prepending every `std` name reference with `std::` is not attractive either. We are attempting to find a safe solution that will not require many dozens of changes (or more) to existing code.

<u>April 2003</u>

In April, Metron continued working on the task of porting DIEMS to Major Shared Resource Center (MSRC) high-performance computers (HPCs). The first platforms targeted were three SGI HPCs. After overcoming some difficulties, we have succeeded in:

1) Building all SPEEDES libraries,
2) Compiling and linking DIEMS, and
3) Running DIEMS and passing the suite of regression tests.

The following sections describe the problems we had to surmount (and the solutions) in order to get DIEMS ported and running correctly. There are two purposes for this: 1) The usual reporting of our activities; and 2) The hope that by documenting these obstacles – and how we circumvented them – future porting will be significantly streamlined. Ideally, someone should be able to load a DIEMS release onto a machine, type a few instructions, and have the software built and running correctly. This is extremely difficult to do because there are so many possible software/hardware configurations in which the software must be built/run, and is compounded by the fact that these configurations are continually shifting and evolving. Nevertheless, this "load-and-play" ideal is a worthy goal to shoot for.

Making SPEEDES Libraries

Although making the SPEEDES libraries with SGI's compiler was straighforward, using the GNU compiler/linker presented some difficult problems. To begin with, the GNU installation on MSRC's SGI machines is haphazard; in particular, one include file needed by DIEMS, `sstream`, is missing. We considered installing g++, but found an easier solution – copying the /usr/include directory from one of Metron's Linux machines to the SGI. Since these were compatible with the g++ version on the SGI (2.95), they worked on the SGI.

The other major problem concerned the use of the GNU linker to build the SPEEDES libraries. This problem arose when using the SPEEDES make system (and matched difficulties reported to Metron by AFRL personnel). Instead of creating the libraries, the linker generated a plethora of "Unresolved symbol" errors and quit. After some investigation, the cause of this problem was determined: because the make system did not provide the "-shared" option in the link command, the linker attempted to create an executable rather than a library. This caused it to try to resolve symbols not defined in the library. Though unfortunate, this was understandable. In the past, when Metron was paid to port SPEEDES to many platforms (including g++ on SGI), the "-shared" option was not available on GNU's SGI linker. Thus, the make system did not use this option in its link command for the GNU/SGI platform. However, because the current GNU linker now *requires* this option, the link command generated by the make system was incorrect.

Speeding up builds on HPCs

This tip was provided by a Metron programmer who has a lot of experience working with HPCs: software builds run much faster when done in one of the "tmp" directories rather than in one's home directory. MSRC's SGI machines provide /var/tmp for this purpose. The reason why builds are faster in /var/tmp is as follows: the tmp directory resides on a disk drive attached to the HPC whereas the home directory is located on a drive connected to the HPC via a networked file system. Thus, the disk I/O on the tmp directory is much faster than that of the home directory. This makes software builds run much faster. The downside is that tmp directories are not backed up (and in fact are occasionally purged of excess data), so important results must be copied back to the home directory, where they are safe.

Building DIEMS

The next task was to compile and link the DIEMS executable. Unfortunately, it was necessary to alter the DIEMS make files in order to do this. One change was caused by the fact (described above) that a g++ include directory had to be copied to the HPC, and therefore had to be designated using an "-I" compiler option. This was caused by a faulty environment, not an error in the DIEMS make system. Another problem caused by the faulty environment concerned the Makefile.in target. Since automake and aclocal were not installed, this rule could not work (and was therefore removed).

A minor irritant is the necessity of always having to remember to use the GNU version of make (gmake) rather than SGI's make. Since the two appear to be grossly incompatible, one is quickly alerted in the event of using the wrong make (since the command usually fails in short order).

Running DIEMS Regression Tests

The first issue to be dealt with concerns the loading of SPEEDES libraries. Since these libraries are not included in the DIEMS executable, they must be loaded at run time. In order to do this, the shell needs to “know” where to find the libraries. This is what the LD_LIBRARY_PATH environment variable is for: to inform the shell where to look for libraries linked to executables. Thus, the SPEEDES library directory must be included in LD_LIBRARY_PATH, as in the following example:

```
setenv LD_LIBRARY_PATH
/lib:/usr/lib:/home/airforce/blank/speedes2.0.1e /delivery/lib/
ArchitectureDirs/IRIX64_IP27
```

We encountered three problems attempting to run the regression test script (RegressionTests.bash). To begin with, this script (and subordinate .bash scripts) must be run with a bash shell, so they begin with "#!/bin/bash". Unfortunately, the bash shell is located in a different place on the SGI HPCs, so every .bash script had to be altered (to begin with #!/usr/gnu/bin/bash).

The next problem was self-inflicted, but other users could be saved some trouble if warned about this pitfall in the DIEMS Installation Guide. When checked out, CVS creates a directory called "diems". This top-level directory name must not be changed, because the RegressionTests.bash script expects it to be called "diems" and will fail otherwise. Because we changed the top-level directory name (e.g. to "apr23Diems"), the RegressionTests.bash script failed.

The third problem concerns the use of the SpeedesServer program by the RegressionTests.bash script. This program needs to be run in order to perform multiple node tests. However, if the location of the SpeedesServer program is not in the user's shell "path" variable, it will not be run and the test will fail. Thus, the user must have the SPEEDES executable directory in his path.

One last quibble about running the RegressionTests.bash script on the SGI. The last step in the script is to run a "TestView.bash" test. This script tests for the existence of the "diemsgraphviewer" program. If it exists, then a test is run; otherwise a message is printed and the script exits. On the SGI HPCs, the script does not identify the fact that there is no "diemsgraphviewer" program and tries to do something (exactly *what* is not clear). The effect is that the RegressionTests.bash script just "hangs" on the last test instead of exiting gracefully. The problem is caused by the following statements in the TestView.bash script:

```
export PROG_GV="diemsgraphviewer"
Result=`which "$PROG_GV" 2> /dev/null | grep -v "no "`
if test -z "$Result"; then
     echo "Diems GraphViewer not found."
     exit
fi
```

Under Linux, the value of the "Result" variable is a string consisting of one space character; this causes the "if test –z” to succeed, resulting in a graceful exit. Under SGI's operating system (IRIX64), Result has the following value:

```
diemsgraphviewer not in /usr/krb5/bin /usr/grd/bin/irix6 .
/bin/usr/bin /usr/brl/bin /usr/ucb /usr/bsd /usr/local/bin ...
```

This causes "if test –z” to fail, which causes the script to continue instead of exiting.

The C++ namespace problem

Last month's report contained a fair bit of discussion about how to deal with the fact that many entities currently in the global namespace will soon be only available in the `std` namespace. We did not have to solve this problem on the SGI platform because of a transition feature in some g++ compilers (e.g. 2.95.2 on the SGI). This feature makes `std` names globally available, so no major changes had to be made in the DIEMS code (e.g. changing all occurrences of `cout` to `std::cout`). In the near future, however, we will have to deal with this situation because g++ 3.0 (and later versions) adheres to the C++ standard, so `std` names such as `cout`, `ostream`, `string`, etc. will no longer exist in the global namespace.

<u>May 2003</u>

Metron investigated solutions to the C++ namespace problem mentioned in previous reports. DIEMS will soon be using the g++ 3.2 compiler, which has moved many common entities (e.g. `cout`, `ostream`, `vector`, `string`, etc.) out of the global namespace and into the `std` namespace. Ideally, it would be preferable not to import all `std` names into the global namespace (via `using namespace std;`) since this increases the possibility of name conflicts. Nevertheless, this is how we will deal with the problem, at least initially, for the following reasons:

1) Impact on DIEMS code

If `std` names are not imported all at once, it will be necessary to insert many instances of `std::` into the code and/or other `using` directives (e.g. `using std::vector;`, `using std::endl;`, etc.). Besides the initial impact, DIEMS programmers will be saddled with the responsibility of dealing with `std` references.

2) Implementation of the upcoming SPEEDES release

For various reasons (like those cited in 1) above), the SPEEDES solution to the namespace problem will be to import all `std` names with the "using namespace std;" directive. This will be part of the g++ 3.2-compatible release of SPEEDES scheduled for this summer. Thus, because some SPEEDES .H files import all `std` names, DIEMS files that include such files will also be importing all `std` names. This would subvert the effort to remove `std` names from DIEMS code. Therefore, it would probably not be wise to attempt this job now. Perhaps in the future, when SPEEDES does not import all `std` names in .H files, it will be worthwhile to purge DIEMS code of unnecessary `std` namespace pollution.

Because we're going to import all `std` names into the global namespace, it is essential that *all* "old-style" include files be removed from DIEMS in order to avoid compiler conflicts. For example, consider a file that does the following:

```
#include <iostream>
using namespace std;

#include "Network.h"
```

If `Network.h` includes the "old-style" file `<iostream.h>`, this will create one or more conflicts that will cause compilation error(s). For example, `<iostream>` defines `std::cout`, which is imported into the global namespace; but, since `<iostream.h>` defines a global object also called `cout`, the compiler cannot know which `cout` is desired and will flag all references to `cout` as "ambiguous". Thus, one should not mix both "old-style" and "new-style" includes. Since the "old-style" files will soon disappear, the only reasonable solution is to remove them and use only "new-style" includes.

On May 14th, AFRL personnel (Mr. Hillman and Mr. Hanna) visited Metron to discuss various DIEMS/SPEEDES issues, including details of porting DIEMS to the SGI HPC, the C++ namespace problem, and the use of SPEEDES features. In the days after this meeting, Metron sent two emails to AFRL: one described in detail the procedure required to port DIEMS (and SPEEDES) to MSRC's SGI HPCs, and the other outlined the old-style/new-style include problem and provided a list of DIEMS old-style includes. Other issues being investigated include: the use of the SPEEDES Data Distribution Management (DDM) system to detect when objects come within a specific range of a given simulation object, and optimizing the SPEEDES GVT parameters. Finally, the make file fix for the SGI (the "-shared" link option) was checked into the official SPEEDES repository so that it will be available in future releases of SPEEDES.

June 2003

AFRL personnel reported that a crash was occurring when running the SpeedesServer together with one SPEEDES node on a given CPU (and running another SPEEDES node on a different CPU). This was a very difficult bug to locate, in part because the problem could not be duplicated on Metron's machines. Nevertheless, after a few days of experimenting (and some help from

Insure's runtime memory checking utility), the cause of the bug was determined and a code fix was implemented and tested. Then a patch was emailed to AFRL. This fix will be integrated into future releases of SPEEDES.

A positive aspect of this was a side benefit that resulted from having to locate and fix this problem. Because it was suspected that memory corruption was the source of the bug, Metron carefully scrutinized much DIEMS code and located a few potential problems and a number of inefficiencies. These were noted and will be fixed.

August 2003

As described in the June report, several problems were discovered in DIEMS code in the course of searching for a bug. The problems all fell into one of two categories: 1) not making events fully rollbackable, and 2) using rollbackable allocation when not necessary. A detailed email was sent to AFRL that described the problems and recommended solutions. Metron also provided phone/email assistance to AFRL programmers with coding problems.

March 2004

Metron spent several hours helping AFRL personnel with a SPEEDES memory leak that was seriously slowing the execution of DIEMS. The problem was located quickly and a patch was emailed to AFRL. After installing the patch, AFRL reported that the problem was repaired.

May 2004

Metron provided assistance to AFRL personnel dealing with two minor problems. The first of these was an inability to send data from DIEMS to an external module at the end time of the simulation. After sending AFRL an email that explained how to do this and provided sample code, the problem was fixed. The second problem was a runtime crash possibly caused by link ambiguities on the SGI 3800. Metron sent a solution addressing the link ambiguity, but has not heard from AFRL whether this fixed the crash.

June 2004

A fair amount of time was devoted to fixing problems in SPEEDES revealed by the attempt to port DIEMS to the SGI 3800 High-Performance Computer. Since this port had been done previously, we did not initially suspect that the problem was in SPEEDES. Thus, we began by making sure that DIEMS was linking to the correct libraries (since there were warnings from the linker that raised the possibility that this might have been the problem).

We soon discovered this was a red herring and began to search for the cause of the problem, a runtime crash. Since this crash could not be duplicated on Intel/Linux platforms, we had to debug via remote connection to the SGI HPC (MSRC's "zornig" machine). This was difficult because of the slow response of the SGI across the internet, and because of the peculiar hodgepodge of software available on the machine. Nevertheless, we were soon able to reproduce the crash in a very simple simulation. Then, with some difficulty, we tracked down the source of the problem: this version of the compiler (g++ v 3.3.2 for SGI) added extra "padding" bytes to the end of a critical SPEEDES message class, thus causing confusion about where certain data was located. While this is completely legal according to C++ standards, it is somewhat unexpected, and had not previously been encountered, despite having ported SPEEDES to many hardware/compiler combinations.

Soon, we created a patch and sent it to AFRL, but the problem persisted. Further investigation revealed that the same problem appeared twice in the code. A fix was tested (with over twenty DIEMS regression simulations) and emailed, but AFRL reported that DIEMS continued to crash in the same manner. This was very puzzling since we felt sure the DIEMS crash had been fixed. We called AFRL personnel and tracked down the problem: after the patch had been inserted and the SPEEDES libraries rebuilt, an installer did not properly copy the new libraries to the "standard" location expected by the DIEMS make system. Thus, DIEMS linked to old libraries and continued to exhibit the same behavior.

The patch will be integrated into the SPEEDES source code repository, and therefore will be available in future releases. But before this can be done, it is necessary to test the new code with our suite of regression tests. We have done this on both the SGI and one of Metron's Intel/Linux machines, but it did not pass all tests. We have determined the reason for most of these, but a couple still remain. The cause of these problems must be determined and fixed before the patch can be integrated into the SPEEDES source code repository. This is currently being done.

July 2004

Last month, we encountered a number of problems revealed by the attempt to port DIEMS to the SGI 3800 High-Performance Computer. Metron provided AFRL personnel with a patch to enable them to proceed with their work on the SGI. However, in the process of testing the patch against our regression tests, a couple of other problems were exposed. It is necessary to resolve such issues before checking the patch code into the SPEEDES repository. Thus, Metron spent time in July resolving the remaining problems on the SGI. A couple of these were minor difficulties having to do with the regression tests and not SPEEDES, but one tough bug remained. After applying some effort, the cause of this problem was located and the code fixed. It turned out to be another instance of the same "byte padding" problem described in June (the fact that internal message objects had four bytes of "padding" appended onto them). Because of this, we searched all SPEEDES source code to ensure that every use of this internal message class was correct. This process is nearly complete and the fixes will be checked in. Also, AFRL will be provided with a patch to fix the problems not already addressed by the patch we sent in June.

In addition to the above, Metron provided AFRL with a number of suggested projects that we might do before the end of the contract in August. After a good bit of negotiation, AFRL decided that Metron would work on an upgrade of the SPEEDES make system so that it uses autoconf and automake, which is a more standard approach to software building/installation. While implementing the autoconf part should not be a problem, it is unclear whether the automake part can be completed in the remaining time. This project has begun, and Metron will make every effort to finish as much as possible before the end of the contract (it is possible that unfinished portions, if any, will be completed with funding from other sources). This upgrade should make SPEEDES easier to install and more portable.

4.0 SPEEDES UPGRADES

In addition to the other two types of activities described, a fair amount of effort was devoted to upgrading SPEEDES. The immediate purpose of this was to support AFRL's efforts; however, these improvements will also benefit government (and other) users of SPEEDES, since it is government-owned software.

Many of these enhancements have already been described. For example, introducing methods for removing an item while iterating through a container class, eliminated a serious gap in the functionality of SPEEDES utilities (two methods were added to each of twelve classes). AFRL programmers working with container classes brought this problem to Metron's attention. Also brought to our attention by AFRL programmers were the bugs described in the previous section. Although SPEEDES is a stable and reliable product, like all software beyond the trivial level, it is not 100% bug-free. However, by fixing the problems reported to Metron by SPEEDES users, these problems are becoming increasingly rare. In the cases of both needed capabilities and bug fixing, Metron has made every effort to be responsive and promptly send a patch so AFRL could continue their work with minimal delay.

Another development project involved a utility to determine good distributions of objects over CPUs used in the simulation. Because event scheduling on the same CPU is much faster than event scheduling across CPUs, performance is improved by placing on the same CPU objects that will schedule a lot of events on each other. This utility would read in the event record from a simulation run and attempt to find a configuration of objects that would improve performance by minimizing inter-CPU event scheduling. This configuration would be specified in a file. SPEEDES would read in this file and distribute the simulation objects accordingly. Unfortunately, funding ran out before this project was complete, and afterwards other tasks took priority over it. However, a good deal of progress has been made, and it may be finished in the future.

The final upgrade to SPEEDES is an improvement to the make system to use autoconf and automake, which is a more standardized approach to software building and installation. Also, this upgrade will result in a more portable and reliable system for building SPEEDES. Although this project began only a couple weeks ago, we have made more progress than anticipated, and are about 80% complete. It is quite possible that this will be finished in the near future and will be available in new releases of SPEEDES.

5.0 CONCLUSION

Over the duration of this contract, Metron has assisted AFRL in developing and debugging DIEMS, in all aspects of integrating DIEMS into an HLA federation, and provided improvements to the SPEEDES framework. Because of the challenging nature of parallel simulation, we firmly believe that support of this kind is an extremely valuable resource. By increasing the rate of development and preventing snags from turning into lengthy delays, it has been a wise investment. We wish to thank AFRL for its confidence and hope that there will be continued collaboration in the future.